1. Для чего используют статические классы?

- Статические классы используются, когда не требуется создание экземпляров класса, и их члены могут быть вызваны непосредственно через сам класс. Они обычно содержат статические методы и свойства.

2. Что может содержать статический класс?

- Статический класс может содержать только статические члены: статические методы, статические свойства, статические поля. Он не может иметь экземпляров.

3. Что такое производный и базовый классы?

- Базовый класс (или родительский класс) - это класс, от которого производится другой класс, называемый производным классом (или дочерним классом).

4. Как используют ключевое слово base?

- Ключевое слово base используется для обращения к членам базового класса из производного класса.

5. В чем заключена основная задача наследования?

- Основная задача наследования - обеспечение возможности использования и расширения функциональности базового класса в производных классах.

6. Может ли объект производного класса иметь доступ к методу basefunc()? Если да, то при каких условиях?

- Да, объект производного класса может иметь доступ к методу базового класса, если этот метод объявлен как protected или public (или internal, если классы находятся в одной сборке).

7. Напишите объявление конструктора без аргументов для производного класса B, который будет вызывать конструктор без аргументов базового класса A.

class B : A

{

public B() : base()

{

// Дополнительная инициализация для класса B, если необходимо

}

}

8. Что такое полиморфизм? Приведите пример.

- Полиморфизм позволяет объектам разных типов обрабатываться единообразно. Пример полиморфизма в C# - использование виртуальных методов и переопределение:

csharp

class Shape

{

public virtual void Draw()

{

Console.WriteLine("Drawing a shape");

}

}

class Circle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a circle");

}

}

// Использование полиморфизма

Shape myShape = new Circle();

myShape.Draw(); // Вывод: Drawing a circle

9. Определите назначение виртуальных функций.

- Виртуальные функции позволяют производным классам переопределять поведение методов базового класса. Это обеспечивает полиморфизм.

10. Кому доступны переменные с модификатором protected?

- Переменные с модификатором protected доступны самому классу, производным классам (наследникам) и классам в том же пакете (или в той же сборке).

11. Наследуются ли переменные с модификатором private?

- Нет, переменные с модификатором private не наследуются и не доступны в производных классах.

12. As, is – что это, как применяется? В чем между ними отличие?

- as используется для преобразования типа с проверкой на null, а is используется для проверки типа без преобразования.

// Пример с использованием as

object obj = "Hello";

string str = obj as string;

if (str != null)

{

Console.WriteLine(str.Length); // Безопасное использование str после преобразования

}

// Пример с использованием is

if (obj is string)

{

Console.WriteLine("obj is a string");

}

13. Поддерживает ли C# множественное наследование?

- Нет, C# поддерживает только одиночное наследование классов, но поддерживает множественное наследование интерфейсов.

14. Можно ли запретить наследование от класса?

- Да, можно использовать ключевое слово sealed для запрета наследования от класса.

csharp

sealed class MySealedClass

{

// Код класса

}

15. Можно ли разрешить наследование класса, но запретить перекрытие метода?

- Да, можно использовать ключевое слово sealed для запрета дальнейшего перекрытия метода в производных классах.

csharp

class MyBaseClass

{

public virtual void MyMethod()

{

// Реализация метода

}

}

class MyDerivedClass : MyBaseClass

{

public sealed override void MyMethod()

{

// Реализация метода в производном классе, и запрет дальнейшего перекрытия

}

}

16. Что такое абстрактный класс?

- Абстрактный класс - это класс, который содержит абстрактные методы (методы без реализации) и может также содержать реализованные методы. Нельзя создать экземпля

р абстрактного класса.

17. В каком случае вы обязаны объявить класс абстрактным?

- Если класс содержит хотя бы один абстрактный метод, то сам класс должен быть объявлен как абстрактный.

18. В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами?

- Абстрактные классы могут содержать абстрактные методы (без реализации), виртуальные классы могут содержать виртуальные методы (с реализацией по умолчанию). Абстрактный класс не может быть создан, а виртуальный может.

19. Какие компоненты класса могут быть виртуальными?

- Методы, свойства, индексы и события могут быть виртуальными.

20. Что такое интерфейс?

- Интерфейс - это контракт, который определяет набор методов и свойств, которые должны быть реализованы классами, поддерживающими этот интерфейс.

21. Что может содержать интерфейс?

- Интерфейс может содержать методы, свойства, индексы и события, но они не содержат реализаций.

22. Как работать с объектом через унаследованный интерфейс?

- Объект можно использовать через интерфейс, как если бы он был экземпляром этого интерфейса. Это позволяет обращаться к методам и свойствам, определенным в интерфейсе.

23. Приведите пример явной реализации интерфейса.

interface IMyInterface

{

void MyMethod();

}

class MyClass : IMyInterface

{

void IMyInterface.MyMethod()

{

Console.WriteLine("Explicit implementation of MyMethod");

}

}

24. Почему нельзя указать модификатор видимости для методов интерфейса?

- Методы интерфейса всегда являются публичными. Модификаторы доступа не допускаются, потому что интерфейс определяет контракт, который должен быть общедоступным для всех классов, реализующих интерфейс.

25. Можно ли наследовать от нескольких интерфейсов?

- Да, C# поддерживает множественное наследование интерфейсов.

26. Назовите отличия между интерфейсом и абстрактным классом.

- Абстрактный класс может содержать как абстрактные, так и виртуальные методы с реализацией. Интерфейс может содержать только методы и свойства без реализации. Класс может наследовать только один абстрактный класс, но реализовывать несколько интерфейсов.

27. Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, IEnumerable?

-

- ICloneable: Используется для создания копии объекта.

- IComparable: Используется для сравнения объектов.

- IComparer: Используется для сравнения двух объектов внешним образом.

- IEnumerable: Используется для поддержки перечисления элементов в коллекции.

28. В какой строке приведенного ниже фрагмента листинга не содержится ошибки?

- К сожалению, фрагмента листинга не предоставлен, поэтому я не могу указать строку без ошибки. Если у вас есть конкретный фрагмент кода, пожалуйста, предоставьте его, и я помогу выявить ошибки.